Architectural Proposal:
Intelligent Guardrails for Data-At-Entry

♦ Overview

Salesforce currently lacks a platform-level mechanism to detect, block, or log toxic, abusive, or harmful text entered into **standard or custom text fields** across any object or surface. While Salesforce provides moderation for community posts, AI prompts, and chat transcripts, the platform does not evaluate the text that users or employees enter into object fields.
Such fields represent the most common and most sensitive entry points in the platform, yet they remain completely unmoderated.

This gap exposes organizations to:
• **Customer → Employee harm:** threats, slurs, or harassment submitted through portals or forms that employees must read as part of their job.
• **Employee → Customer harm:** harmful or abusive text entered by staff into customer-facing fields that later appear on invoices, emails, statements, or portal pages.
• **Brand safety risk:** reputational damage, legal exposure, compliance failures, and employee/customer protection.
• **AI Bad actor → Systemic harm:** automated bots injecting toxic content at scale, polluting the System of Record and exposing the brand to widespread liability.

♦ Documented Incidents

Comcast Billing Name Incidents (2015) *Customer billing names were changed to offensive slurs ("A**hole Brown", "Super B**** Bauer"), resulting in public apologies, refunds, and national press coverage. These were simple* ***text field edits*** *— exactly the surface Salesforce does not moderate.*

Hostile Work Environment Case (2023) *Legal filings described slurs and abusive language contributing to a hostile work environment. This highlights the employee-safety dimension: harmful text can be submitted by customers or coworkers, and employees are forced to read it as part of their job.*

Although these incidents did not occur on Salesforce: any system that accepts free-form text is vulnerable to harmful or abusive content unless moderation is built in at the platform level. Salesforce is often the system of record for the same categories of user-generated text, making it the logical place for native safeguards.

♦ Current Workarounds and Observed Limitations

Salesforce clients already try to handle harmful text in various ways, but the solutions they build are localized, require custom coding, and fail to scale. Based on research and platform experience, bellow are the primary workarounds used today. The approaches listed here are either incomplete, unreliable, or expensive, and none of them provide consistent, platform-level moderation ***specifically for text fields*****.**

- **External API callouts**
  This approach calls external toxicity or profanity APIs from Flow or Apex. It burns API usage limits and depends on an outside service being fast and available. This approach is expensive and, eventually, unrealistic to run on every save as volume grows.

- **Validation Rules with REGEX or CONTAINS**
  This approach only works on exact text patterns and doesn't provide any real understanding of the content. It also does not fit well with implementations of Digital Experiences (Experience Cloud) when expectations are to apply validations on the client side at the UI level. This workaround also does not scale across objects and does not offer any logging or governance.

- **Custom Flows or Apex triggers**
  This approach implements inline checks directly in Apex triggers or formula logic in flows. These typically rely on hardcoded keyword lists or custom metadata that require continuous updates. While the validation logic itself can be reused, each Flow or trigger must be wired to call it, which becomes difficult to maintain as the number of fields and automations grows. This approach provides no governance, logging, or contextual understanding of the content.

- **Manual review**
  This approach relies on employees (managers) to manually check notes or comments. It is slow, inconsistent, reactive, not proactive, and does not prevent employees from being exposed to harmful content in the first place.

- **Scheduled custom scripts**
  This approach runs scheduled Apex or external scripts to periodically scan records and identify toxic field values. It only detects harmful content after it has already been saved and potentially viewed, and it cannot block harmful input in real time.

- **Localized AI prompts via Prompt Builder**
  This approach uses Prompt Builder to run toxicity checks through the Einstein Trust Layer, providing high accuracy. However, it remains a localized workaround: each implementation must be manually wired into a Flow or trigger and duplicated across objects. Crucially, it does not provide native, centralized logging or enforcement out-of-the-box. To achieve full audit and governance, an organization must build and fund a custom logging architecture on top of the AI prompt, making the solution either costly or architecturally incomplete.

PROPOSED SOLUTION

Salesforce should introduce a native, platform-level capability for evaluating toxic content in text fields, delivered with Salesforce-style configurability (org-level, object-level, and/or field-level) to let each organization determine where moderation applies.

♦ The Core Concept

If a company decides that an Agentforce bot should block an order containing racial slurs, why shouldn't that same rule apply when the order is submitted via a standard field form? Today, the "brain" exists for the bot, but "push back" is missing at the input field level.
To close this gap, Salesforce can reuse the same reasoning intelligence that is already built into Agentforce to the act of data entry itself.

To achieve this, the platform should provide a **Unified Agentic Reasoning Layer**. This extends Salesforce's agentic capabilities from conversations (Agentforce Chats) into system-of-record entry points (Form Fields). Whether an order is submitted via a conversational bot or a standard text field, the underlying reasoning capability remains the same.
This creates a **Unified Agent Layer for Interaction and Data-At-Entry**, ensuring that "Invisible Agent" protects the System of Record before the final commit. (Real-time sync option). The service could also evaluate content in near-real time to block, flag, log, and/or track harmful text. (Data Load, async option)
By treating the AI as a universal decision layer, Salesforce ensures that policy enforcement is consistent regardless of the interaction model.

♦ Native Platform Integration

The proposed solution integrates a native toxic context scan directly into the **save text-field process**. This pipeline leverages the **Einstein Trust Layer** to evaluate submitted text, applying admin-defined enforcement rules in real-time.

- **Declarative Configuration**
  The concept is intentionally designed to align with Salesforce's declarative, metadata-first philosophy. By exposing this capability through established platform patterns—such as field attributes, object settings, and org-wide policies. Admins can enroll fields into a managed security perimeter as easily as they designate PII or encryption. This ensures that the user experience remains familiar and consistent with any other type of field security configuration

- **Language-Agnostic**
  The architecture uses native multilingual classifiers, requiring no extra setup for global organizations.

- **Governance and Workflow**
  Review workflows are intentionally left to the customer. Salesforce provides the audit log and permission-controlled visibility, while organizations use their existing Case Management, Standard Objects, HR workflows, or custom processes to review and act on flagged content. This **avoids forcing a one-size-fits-all moderation** queue and keeps the feature aligned with Salesforce's metadata-first, admin-controlled philosophy.

- **Einstein Analytics**
  Organizations can analyze moderated events to identify false positives, monitor employee exposure levels, and track behavioral patterns from external users. This ensures the security perimeter evolves with the business.

♦ Configuration / Enforcement Approaches

While Salesforce's internal implementation details aren't public, the platform's existing patterns offer enough direction to outline realistic approaches. The suggestions here rely only on visible platform capabilities and established patterns, aiming for consistency with how similar features work today. A more optimal or deeply integrated approach may exist and determine that is ultimately up to Salesforce as the platform owner, with full knowledge of its architecture, capabilities, and internal design conventions.

• **Org-Level Enablement** Applies toxicity screening across all objects and all text-type fields.
Similar to:

Org-wide Duplicate Management activation (feature turned on once, then rules apply where configured)
Einstein Trust Layer policies at the tenant level (central AI governance applied across multiple surfaces)
Agentforce activation at org level (platform-wide enablement that unlocks AI capabilities across experiences, even if individual teams adopt it unevenly)

• **Object-Level Enablement** Applies screening to all text-type fields on a specific object (standard or custom).
Similar to:

Object permissions in Permission Sets and Profiles (object-level configuration that requires all text fields on object to be subjects of moderation)
Duplicate Rules configured for a specific object with conditions (object-level rule that evaluates every save, but only enforces when its conditions are met)
Object-level settings and behaviors (features that are enabled or configured per object, such as search layouts, compact layouts, or object-specific actions)

• **Field-Level Enablement** Applies screening to individual text fields only.
Similar to:

Field Attributes (checkboxes used to define field behavior and compliance at the point of creation, such as "Required," "Unique," or "PII/Compliance")
Field-Level Security (mark profiles that may bypass or required toxic check)
Permission-set–based field access (allows to set narrow list of fields that are subject to check per specific profiles)

• **Same-Time Evaluation and Blocking** Evaluates text during the save operation and can block, warn and/or clean text based on policy, when logging or tracing is not required.
Similar to:

Duplicate Rules (Block / Allow with Alert)
Validation Rules

• **Logging, Analytics, and Behavior Signals** Captures events, patterns, and risk indicators for internal/external users.
Similar to:

Shield Event Monitoring
Field Audit Trail
Einstein Trust Layer telemetry and policy logs

♦ Performance and Scalability Considerations

- **Execution Modes**

To maintain platform stability and minimize latency during save operations, the moderation service should support two processing modes:

- **Synchronous Enforcement (Real-Time):** Used for customer-facing surfaces or high-risk fields where harmful content must be logged, flagged and blocked ***before*** it enters the system of record.

- **Asynchronous Auditing (Near Real-Time):** Used for internal operations, high-volume data dumps, or legacy integrations (e.g., nightly ingestion of parsed physical forms). Following Salesforce's own best practices for high-volume ingestion—where synchronous validations are often bypassed for performance—this "Invisible Agent" evaluates the text in the background. This ensures the system is scrubbed, flagged, or masked before any human employee is exposed to the data.

- **Reliability and Data Integrity**

  To ensure enterprise-grade uptime and prevent "Race Conditions," the architecture includes:

  - **Fail/Latency Path**: To ensure zero disruption to core business operations, the moderation service operates on a 'Fail-Open' policy. If the AI service experiences latency or downtime, the transaction completes normally, and the records are automatically flagged for retrospective **asynchronous auditing** once the service is restored.

  - **Data Integrity**: The "Clean Bucket" Logic For high-consequence, public-facing fields, the system adopts a "Semantic Malware" approach. Just as files are scanned before being promoted to a server (AWS S3 Bucket analogy), these fields are treated as Pending until cleared by the Invisible Agent. This eliminates the "Race Condition" where toxic data could be committed and immediately used in an automated downstream process (like an invoice). By marking specific fields as Requires Safety Clearing, organizations can ensure that high-liability data remains quarantined until it is verified as "Clean."

- **Performance and Tradeoffs**

  Performance issues is proposed to use specialized modeling:

- **SLM (Small Language Model) or a dedicated Toxicity Classifier.** These are significantly faster than average LLM because their main task will be to identify toxic content, not to be a “know-it-all" tool.

The slowness of the “Submit” button is a **calculated tradeoff:** it’s cheaper than a million-dollar lawsuit. (Ask your Legal department). Just as Salesforce accepts the performance overhead of Duplicate Rules and Validation Rules to ensure data integrity, organizations will prioritize a latency hit over the catastrophic costs of brand damage and lawsuits. In the enterprise world, reputation and workplace safety are far more expensive than a few extra milliseconds on a save operation.

♦ Security and Privacy Considerations

- **No Data Leaks:** Everything stays inside the Salesforce Trust Layer.

- **Intent, Not Censorship:** This isn't a "bad word" shame. The goal is to catch harmful intent while protecting the business's right to speak. That’s why masking options are suggested. (**safety,** not

surveillance)

- **Audit Stamp:** Every flag creates a log. This protects the employee from being "AI-accused" by a hallucination or false positives and eventually industry-specific driven.

- **Need-to-Know Access:** Only those with the right Permission Sets aligned with organization policies (Legal, HR) can see what was caught. The rest of the org just sees a clean system.

♦ Audit Trails and Industry Context

This proposed solution offers Salesforce a unique opportunity to receive structured feedback for business-specific governance. It allows organizations to identify new patterns to refine prompts for their specific business context—improving AI accuracy over time by closing gaps and eliminating false positives.

Industry-Specific Refinement: This framework allows Salesforce to evolve from a generic filter to specialized, industry-specific classifiers. By shipping pre-tuned models within **any Cloud**, Salesforce can offer a "contextual lens" that distinguishes between a clinical report and a safety threat, ensuring moderation aligns with the line of business.
By using the **Audit Trails** as a feedback loop, Salesforce doesn't have to guess what "Industry-Specific Toxicity" looks like. It can aggregate (anonymized) data from the first 500 customers to define the industry standard.

Example of different contextual lenses per industry:
- Education Cloud
*Student types in Case.Description: "My friend wants to **commit suicide**"*
***Action:** High-priority Flag/Alert. This is an urgent safety signal for a counselor.*
- Health Cloud
*Doctor types in Case.Description: "Patient tried to **commit suicide**"*
***Action:** Allow/Log. Clinical documentation.*

♦ Salesforce Platform Benefits

- **Harmonized governance**

This proposed solution offers Salesforce a unique opportunity to receive structured feedback for industry-specific governance. By categorizing "audit stamps" across different industries—whether it's Government, Education, or Retail—Salesforce can harmonize governance just as it harmonizes data. While the framework might start as a generic "one-size-fits-all" filter, as organizations refine their own toxicity model**s** based on real-world patterns, it becomes a strategic data asset, which Salesforce can **package as industry-specific governance models**.

- **The Shield for Human Input**

This proposal offers Salesforce a short-window opportunity to define the industry standard for Input Security. While competitors like Microsoft and ServiceNow move toward "Safer Enterprise Workflows" with integrated toxicity classifiers, the Salesforce ecosystem remains vulnerable at **Data-At-Entry** surface.
By treating free-form text fields not just as data points, but as active liability surfaces, Salesforce can bridge the gap between CRM and Cybersecurity. Establishing this native Data-at-Entry layer closes a

critical competitive gap and prevents rivals from claiming the "**Safer CRM**" title - all while protecting the brand's reputation and the employee's workplace safety.

♦ Closing Statement

Traditionally, cybersecurity focuses on three states:
Data-at-Rest (encryption)
Data-in-Transit (APIs)
Data-in-Use (processing).
This proposal identifies the unshielded and underestimated frontier: **Data-at-Entry**.

By establishing this fourth security state, Salesforce can offer the first truly end-to-end trust environment:
Protecting the Brand: Neutralizing "Headline Risks"—like the Comcast billing incidents—before they trigger public apologies and million-dollar lawsuits.
Protecting the Data: Ensuring the System of Record remains clean, professional, and compliant.
Protecting the Human: Safeguarding employees from the emotional harm of workplace toxicity, slurs, and threats.

If harmful human-to-human interaction is not a strong enough argument, in the modern era, **AI-to-Human harm** has become a feasible reality. Uncontrolled bots can now generate and inject harmful content at a volume that manual moderation cannot touch.
Data-at-Entry protection needs to be shielded before the "*System of Record*" becomes a "***System of Liability***".

(mic drop 🎤)